Lab assignment-2.5

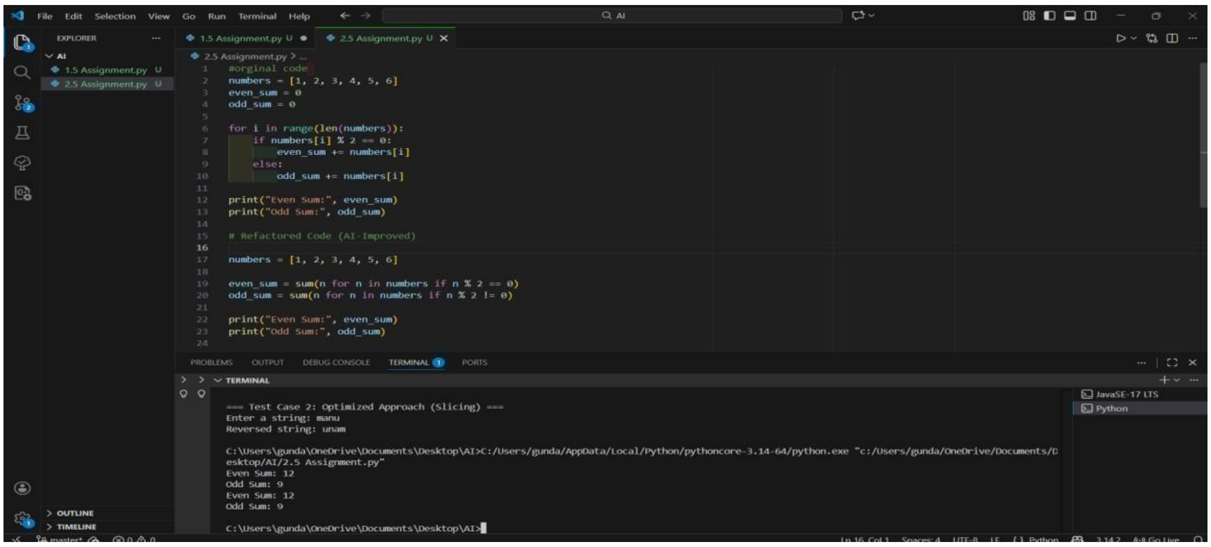
Y. Sahasra

2303A51269

Task -1: Refactoring Odd/Even Logic (List Version)

Prompt : Write a Python program to calculate the sum of odd and even numbers in a list Code

and output :



```python
#orginal code
numbers = [1, 2, 3, 4, 5, 6]
even_sum = 0
odd_sum = 0

for i in range(len(numbers)):
    if numbers[i] % 2 == 0:
        even_sum += numbers[i]
    else:
        odd_sum += numbers[i]

print("Even Sum:", even_sum)
print("Odd Sum:", odd_sum)

# Refactored Code (AI-Improved)

numbers = [1, 2, 3, 4, 5, 6]

even_sum = sum(n for n in numbers if n % 2 == 0)
odd_sum = sum(n for n in numbers if n % 2 != 0)

print("Even Sum:", even_sum)
print("Odd Sum:", odd_sum)
```

```
=== Test Case 2: Optimized Approach (Slicing) ===
Enter a string: manu
Reversed string: unam

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

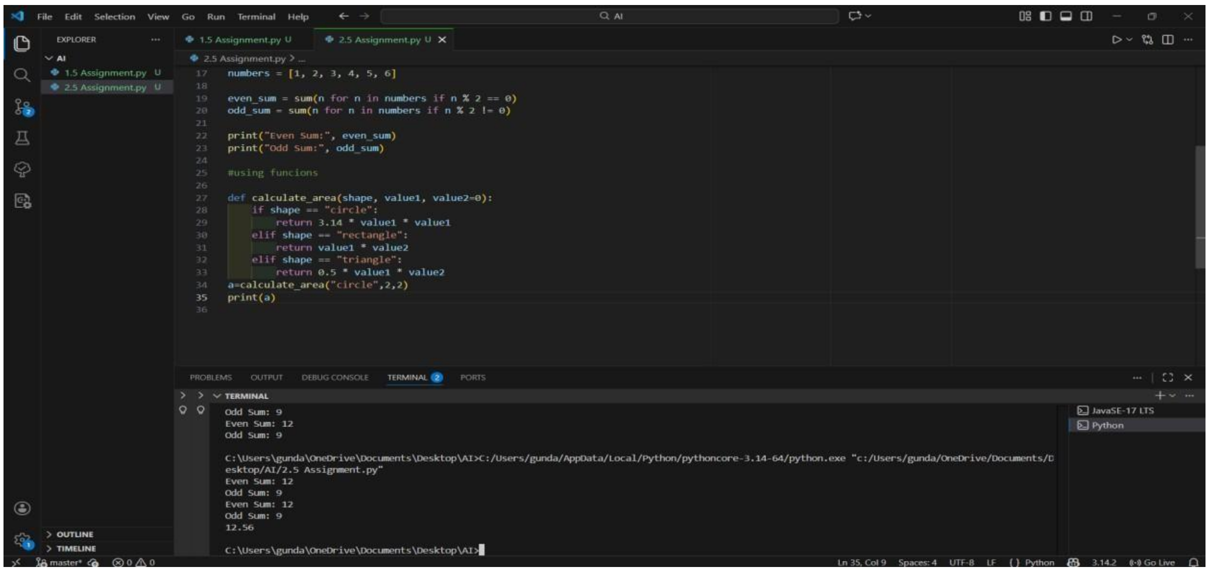
Explanation

The refactored code is shorter, more readable, and efficient.
It removes manual loops and uses Python's built-in sum() with conditions, making the code easier to maintain.

Task 2: Area Calculation Explanation

Prompt : Explain a Python function that calculates the area of different shapes Code

and output :



```python
numbers = [1, 2, 3, 4, 5, 6]

even_sum = sum(n for n in numbers if n % 2 == 0)
odd_sum = sum(n for n in numbers if n % 2 != 0)

print("Even Sum:", even_sum)
print("Odd Sum:", odd_sum)

#using funcions

def calculate_area(shape, value1, value2=0):
    if shape == "circle":
        return 3.14 * value1 * value1
    elif shape == "rectangle":
        return value1 * value2
    elif shape == "triangle":
        return 0.5 * value1 * value2
a=calculate_area("circle",2,2)
print(a)
```

```
Odd Sum: 9
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9
Even Sum: 12
Odd Sum: 9
12.56

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

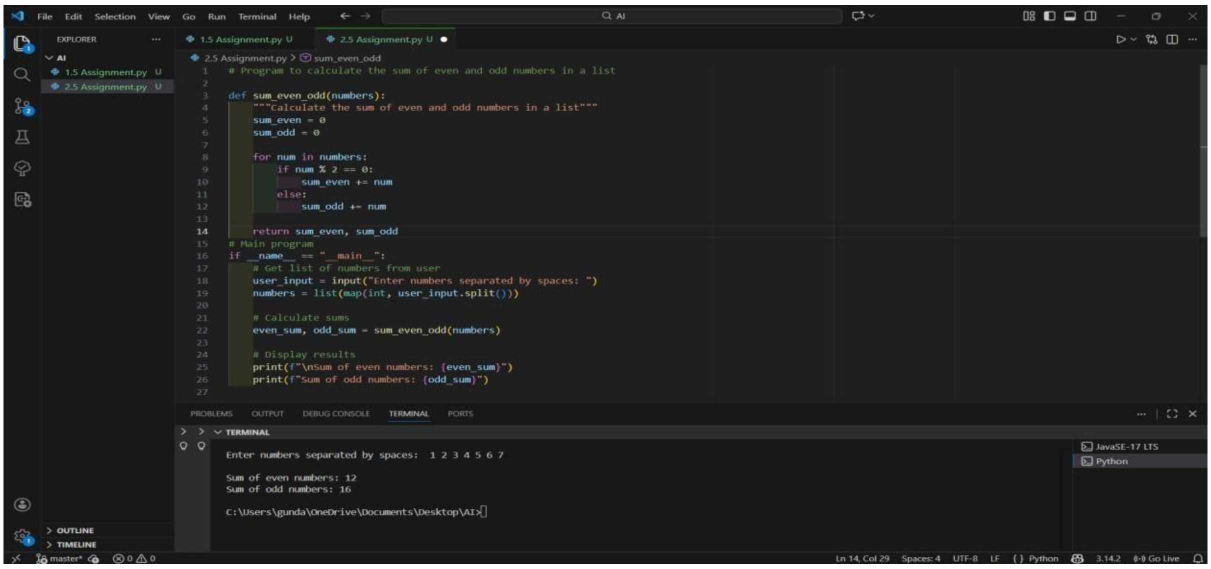
Explanation Gemini clearly explains how the function works for different

shapes.

It describes the parameters, logic flow, and formulas used, which helps beginners understand the code easily.

Task 3: Prompt Sensitivity Experiment

Prompt 1: Write a Python program to calculate the sum of even and odd numbers in a list



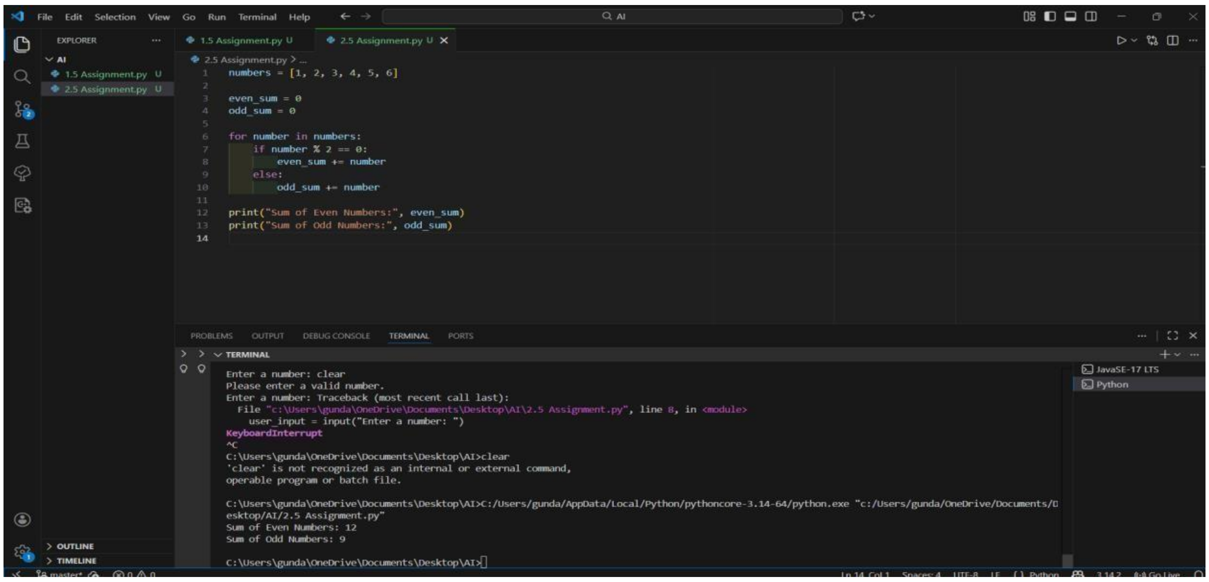
```python
# Program to calculate the sum of even and odd numbers in a list

def sum_even_odd(numbers):
    """Calculate the sum of even and odd numbers in a list"""
    sum_even = 0
    sum_odd = 0

    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num

    return sum_even, sum_odd
# Main program
if __name__ == "__main__":
    # Get list of numbers from user
    user_input = input("Enter numbers separated by spaces: ")
    numbers = list(map(int, user_input.split()))

    # Calculate sums
    even_sum, odd_sum = sum_even_odd(numbers)

    # Display results
    print(f"\nSum of even numbers: {even_sum}")
    print(f"Sum of odd numbers: {odd_sum}")
```

```
Enter numbers separated by spaces:  1 2 3 4 5 6 7

Sum of even numbers: 12
Sum of odd numbers: 16

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

Explanation:

For **Prompt 1 (Basic Prompt)**, Cursor AI generated a simple loop-based program using conditional statements. This version is easy to understand and suitable for beginners, but it uses more lines of code and manual variable updates.

Prompt 2: Write a clean and readable Python program to find the sum of even and odd numbers in a list suitable for beginners Code and output:

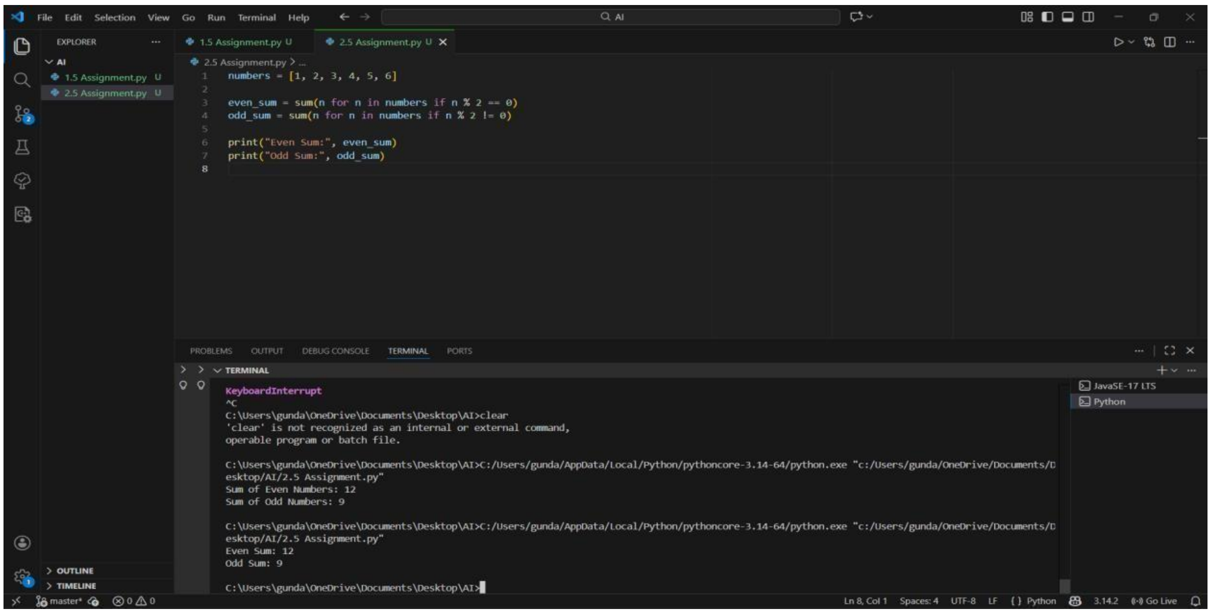


```python
numbers = [1, 2, 3, 4, 5, 6]

even_sum = 0
odd_sum = 0

for number in numbers:
    if number % 2 == 0:
        even_sum += number
    else:
        odd_sum += number

print("Sum of Even Numbers:", even_sum)
print("Sum of Odd Numbers:", odd_sum)
```

```
Enter a number: clear
Please enter a valid number.
Enter a number: Traceback (most recent call last):
  File "c:\Users\gunda\OneDrive\Documents\Desktop\AI\2.5 Assignment.py", line 8, in <module>
    user_input = input("Enter a number: ")
KeyboardInterrupt
^C
C:\Users\gunda\OneDrive\Documents\Desktop\AI>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

Explanation : For **Prompt 2 (Readability-Focused Prompt)**, the AI produced code with clearer variable names and better formatting. Although the logic is similar to the basic version, readability and clarity were improved, making the code easier to review and maintain.

Prompt 3: Write an optimized Python program to calculate the sum of even and odd numbers in a list using built-in functions Code and output:

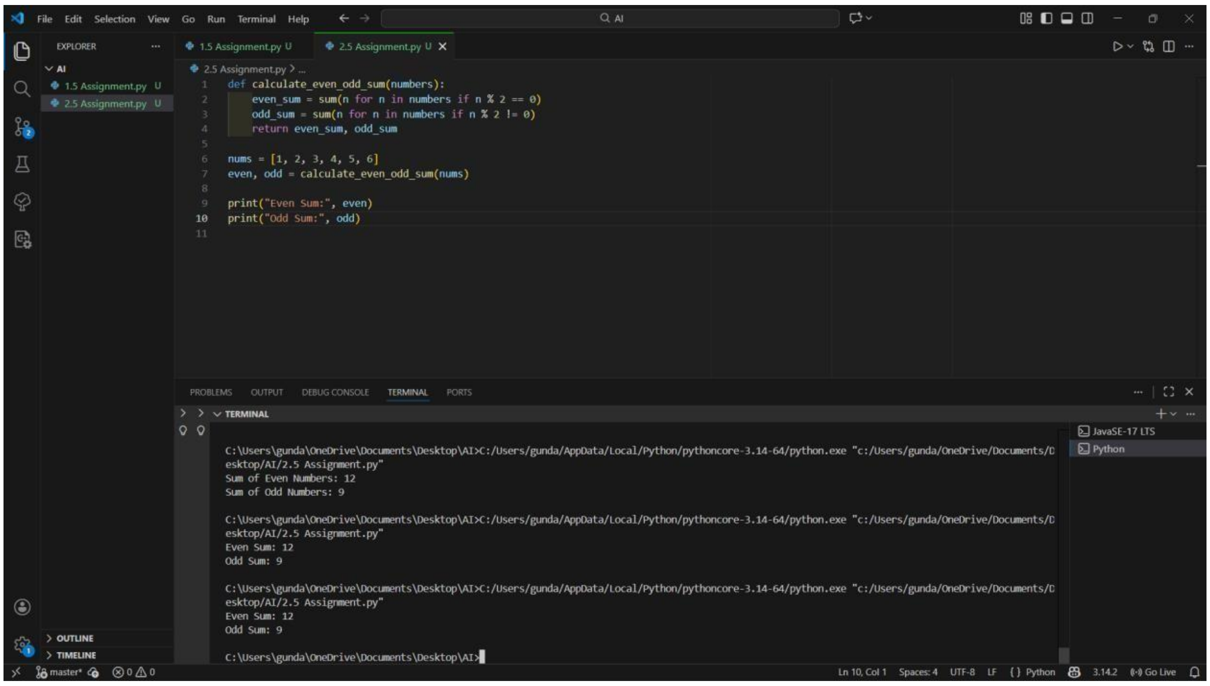


```python
numbers = [1, 2, 3, 4, 5, 6]

even_sum = sum(n for n in numbers if n % 2 == 0)
odd_sum = sum(n for n in numbers if n % 2 != 0)

print("Even Sum:", even_sum)
print("Odd Sum:", odd_sum)
```

```
KeyboardInterrupt
^C
C:\Users\gunda\OneDrive\Documents\Desktop\AI>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

Explanation: For **Prompt 3 (Optimized Prompt)**, Cursor AI generated a more efficient solution using Python's built-in sum() function along with conditions. This version reduced the number of lines and improved code efficiency while maintaining correctness.

Prompt 4 : Write a Python program to calculate the sum of even and odd numbers in a list using functions Code and output :



```python
def calculate_even_odd_sum(numbers):
    even_sum = sum(n for n in numbers if n % 2 == 0)
    odd_sum = sum(n for n in numbers if n % 2 != 0)
    return even_sum, odd_sum

nums = [1, 2, 3, 4, 5, 6]
even, odd = calculate_even_odd_sum(nums)

print("Even Sum:", even)
print("Odd Sum:", odd)
```

```
C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

Explanantion:
For **Prompt 4 (Function-Based Prompt)**, the AI created a modular solution using a user-defined function. This approach improves reusability, debugging ease, and maintainability, making it suitable for larger applications.

Task 4: Tool Comparison Reflection

Reflection

Based on the experiments performed in this lab, Google Gemini, GitHub Copilot, and Cursor AI each have different strengths.
Google Gemini is very useful for understanding code, as it provides clear explanations and works well in Google Colab, especially for beginners.
GitHub Copilot offers real-time code suggestions inside VS Code and is best suited for daily development and writing production-ready code.
Cursor AI is effective for experimenting with different prompts, refactoring code, and analyzing multiple coding approaches.